A Cross-Domain Experiment in Case-Based Design Support: ArchieTutor

Ashok Goel*
College of Computing
Georgia Institute of Technology

Ali Malkawi
College of Architecture
Georgia Institute of Technology

Michael Pearce
Galaxy Scientific Corporation
Atlanta, Georgia

Kim Liu
College of Computing
Georgia Institute of Technology

Abstract

ArchieTutor is an experiment in combining case-based and multimedia technologies to support the teaching of architectural design. It provides an architectural student with access to a library of past design cases, and enables the student to retrieve and browse through cases that are similar to a given design problem. The cases are annotated with design lessons learned from them. In addition, the cases are explicitly related to relevant domain models, and design principles and guidelines. The system uses these principles, guidelines and models to explain, justify or critique a design choice, and to elaborate on a design lesson. In addition, it uses cases to illustrate a given principle, guideline or model.

Introduction

Over the last few years, our research group has developed a series of case-based design support systems. Each system in this series represents an experiment in combining case-based and multimedia technologies for supporting conceptual design in complex domains. In the Archie project [Pearce *et al.* 1992], we explored the use of design cases for aiding architects in designing new office buildings. In the AskJef project [Barber *et al.* 1992], we investigated the use of multiple types of knowledge, including design cases, for advising software engineers on the design of human-machine interfaces. The more recent ArchieTutor system represents a cross-domain experiment in using AskJef's framework for supporting design teaching in Archie's domain.

*Acknowledgements: This paper has benefited from many discussions with Janet Kolodner. It has been supported in part by the Office of Naval Research (contract N00014-92-J-1234). Contact: Ashok Goel, College of Computing, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, Georgia 30332; email: goel@cc.gatech.edu; phone: (404) 853-9371.

The goal of this paper is to briefly sketch our evolving model for case-based design support, and to outline its current form as instantiated in ArchieTutor. We begin with descriptions of Archie and AskJef, focusing on the lessons we learned from them. We continue with a description of ArchieTutor.

Archie

We designed Archie with the idea of using computer-based case libraries to support human decision making [Kolodner 1991]. The system aids architects in conceptual design of new office buildings [Goel *et al.* 1991]. It provides architects with a library of past design cases to help them in two subtasks of conceptual design: design generation and design critiquing. In the generation phase, after an architect specifies the goals of a new design problem, Archie uses the goals to probe its case library and retrieves design plans corresponding to similar goals. In the critique phase, when an architect proposes a specific design to achieve the design goals, the system first uses domain models to determine the potential outcomes of the design, and then uses these outcomes to probe its case library and retrieve designs that resulted in similar outcomes. By looking at the outcomes of the retrieved cases, the architect can predict the outcome of the proposed design.

The memory organization of Archie is primarily case-based, with models providing indexes to design goals and outcomes. A design case in Archie packages specifications of the goals, plans and outcomes of a specific design. Case features are organized hierarchically, and include features relating to the organization that uses the building, the core (permanent structure) of the building, the partitions that separate the space in the building, and the furniture. The domain models in Archie specify the system's knowledge of qualitative relations between features characterizing office buildings, and are used to critique a proposed design. For example, if an architect wanted to critique a design from the viewpoint of lighting quality, Archie would use the lighting quality model to predict that the proposed design will result in high glare intensity (and thus low lighting quality), and use this result to retrieve past designs that resulted in similar outcomes.

Archie contains twenty cases and six models. The system was built using Cognitive Systems' Remind tool on a MacIntosh II. It was evaluated only informally, mainly by showing it to some domain experts.

Lessons Learned

The Archie project was a mixed success and we learned much from its limitations [Pearce *et al.* 1992]:

Design cases are useful for both design generation and critiquing. Cases can be a source of new design concepts and ideas. They can also provide parts of the desired design solution. However, Archie does not force (or even assist) the architect to use the retrieved case in any way. Instead, it provides concrete examples that prompt the architect to think about relevant design issues. In addition, cases can provide useful design lessons learned from past designs in the form of problems encountered in them and remedies to fix the problems. This knowledge is potentially useful for critiquing proposed designs and correcting problems in them.

Domain models are useful for design critiquing. The domain models in Archie enable the system to critique proposed designs, and to find possible conflicts in or problems with proposed design plans. Each model in the system acts as an architectural specialist that can evaluate a design from its own point of view and point out problems in the design.

Design cases and domain models are useful for design cooperation. A computer-based case library acts as a shared external memory. By including enough knowledge about the design goals, plans, outcomes and lessons in its cases, Archie provides the designer access to the work of previous architects. Similarly, by providing access to the perspectives of various domain experts via the domain models, the system enables the designer to anticipate and accommodate their views on the evolving design.

Design cases are large and incomplete. The amount of knowledge necessary for design in a non-trivial domain is very large. This raises the issue of decomposing design cases into smaller, more useful subcases. In addition, knowledge of real-world design cases often is incomplete. This in turn raises the issue of how to use incomplete and uncertain case knowledge to support human decision making.

Multiple types of knowledge are required for design support. Although Archie contains domain models in addition to design cases, the user does not have access to these models. Design problem solving however appears to involve several types of knowledge in addition to that of design cases. For example, design principles and guidelines are needed to generate new designs, and domain models are useful in critiquing proposed designs.

Usability issues are very important. The emphasis of our work on Archie was on collecting, analyzing, indexing and entering knowledge of architectural design cases into the system. We did not pay equal attention to Archie's interface, and ignored the usability and learnability issues that are important to end users. As a result, its interface is usable by knowledge engineers but not by architects.

AskJef

The AskJef project picked up where Archie left off. AskJef supports software engineers in designing human-machine interfaces [Barber *et al.* 1992]. It provides software engineers with access to a computer-based library that contains several types of knowledge: (i) design cases from the domain of human-machine interface design, (ii) design stories from other domains, (iii) design principles and guidelines, (iv) design objects, and (v) prototypical designer errors. The different types of knowledge in the system are cross-indexed through the annotations on design cases. The case annotations point out the good and bad features of the interface, and are linked to relevant guidelines and principles. The design guidelines and principles help to further explain the case annotations, and the design cases and stories help to illustrate the guidelines.

AskJef employs text, graphics, animation and voice to help software engineers understand both the general domain of interface design and specific problems in the domain. It contains about ten design cases and forty stories, half dozen design principles and forty guidelines, dozen prototypical errors and twenty interface objects.

AskJef's memory was constructed using Inference Corporation's ART-IM knowledge representation toolkit and its interface was designed using Asymetrix ToolBook on an

IBM compatible PC. Various versions of the system were demonstrated to domain experts through out its development, and their feedback was incorporated in the succeeding versions. Software engineers were invited to use the final version of the system for a variety of design and redesign tasks, and extensive data on their use of the system was collected and analyzed.

Lessons Learned

Multiple types of knowledge are needed for design advising. From our experience with Archie, we knew that cases were not the only type of knowledge useful for the design of complex systems. Experiments with AskJef confirmed this hypothesis: designers need access to different types of knowledge at different stages in the design process. This knowledge is necessary for the user to organize case knowledge in a useful way. For example, design principles provide abstract knowledge that allows the user to generalize across design cases.

Different types of knowledge need to be cross-indexed. The user should have access to the different types of design knowledge stored in the system's memory. Hence, the different types of knowledge in the system's memory need to be cross-indexed so that the user can easily navigate the memory and move from one type of knowledge to another. This navigation approach gives the user access to all of the knowledge in the system but constrains its presentation so that the user is not shown an incomprehensible collection of information at any time.

The history of a design provides both design rationale and case decomposition. In AskJef, a design case is decomposed into several versions that together form a chronological history of the interface design. The system presents only one version at a time, making it easier for the user to "digest" the information stored in a case. In addition, the temporal decomposition of a design provides a rationale for the interface design: the user can look at the evolution of the design, the problems with each version, and the subsequent solutions to interface problems.

The interface organization should reflect the memory organization. One of the important lessons we learned from experiments with AskJef is that the organization of the display presentation should reflect the organization of knowledge in memory. This helps users in building a mental model of the system in the terms of the relationships between different types of knowledge and the functional roles they play. This mental model in turn makes it easier for them to use the system.

Case libraries are useful for design tutoring. Another important lesson we learned from AskJef is that the system is useful not just for aiding design but also for teaching design. Experiments with AskJef indicate that the system is useful for design teaching in several ways. First, the annotated design cases help the inexperienced software engineer understand a common method for generating design solutions, and, similarly, the design principles and guidelines help the software engineer understand a common method for critiquing proposed solutions. Second, the annotated design cases and stories, the design principles and guidelines, the domain objects, and the relations among them, help the inexperienced software engineer in understanding the nature of the domain of interface design as well as the structure of design problems and solutions in the domain. Third, the design rationales and the prototypical errors help the inexperienced software engineer in understanding typical problems with interface designs and ways of fixing some of these problems.

ArchieTutor

In a new cross-domain experiment, we are presently applying AskJef's knowledge and interface organization ideas to Archie. The new system, called ArchieTutor, operates in the domain of architecture, as does Archie. However, instead of supporting professional architects in solving complex design problems, our goal for ArchieTutor is to support design teaching in beginning architectural classes. More specifically, the system is intended to support design teaching in two ways. First, building on the results of AskJef, it is intended to support design teaching by helping beginning architectural students in understanding the nature of the design domain of office buildings, and the structure of design problems and solutions in the domain. Second, following the results of AskJef and using Archie as a base, it is intended to support design teaching by exposing students to some of the knowledge sources and skills useful in design generation and design critiquing.

Knowledge, Memory, and Interface

Like AskJef, ArchieTutor uses multiple types of knowledge: (a) design cases, (b) domain models, and (c) design principles and guidelines. The organization of knowledge in ArchieTutor's memory also is similar to that in AskJef, but ArchieTutor makes the relationship between the different types of knowledge more explicit than AskJef. Figure 1 illustrates these relations. The student has access to all knowledge in the system's memory. The design cases, design principles and guidelines, and domain models are cross-indexed in such a way that the student can navigate from any one type of knowledge to another.

Following the results of the AskJef experiment, the organization of ArchieTutor's interface strongly reflects the organization of knowledge in its memory. Its interface consists of four major screens. The first screen is for specifying the design problem at hand. The other three screens are for presenting the different types of knowledge in the system, as described below.

Design Cases: The content, representation and internal organization of a design case in ArchieTutor is similar to that in Archie. As in Archie, a case in ArchieTutor contains knowledge of the design goals, plans and outcomes of the case. This knowledge is organized at several levels of abstraction including the type of organization using the office building, and the core and partitions of the building. Again, as in Archie, a case in ArchieTutor is annotated by different kinds of annotations, including the good and design features of the case and the lessons learned from it.

Figure 2 illustrates the presentation of a case in ArchieTutor. The graphical representation of the case allows the student to focus on a given chunk of the design by zooming in, or look at the design as a whole by zooming out. The organization of the screen allows the student to review the case annotations. A graphical pointer on the case display relates the text annotations to a graphical representation of the case. In addition, the pointer relates the case annotations to relevant domain models, and design principles and guidelines. The student can at any time choose to browse any one of these and review their structure in more detail. The case presentation screen also lists other cases relevant to the given design problem, and allows the student to view them.

As indicated in Figure 2, cases in ArchieTutor differ from those in Archie in two ways. First, cases in ArchieTutor are cross-indexed with its domain models and design guidelines. This cross-indexing is done through the annotations on the cases: an annotation on a case may appeal to a domain model or a design guideline for elaboration. Second, this cross-indexing requires an enhanced indexing vocabulary. Archie's cases are indexed by the design goals they satisfy and by the outcomes of the design plans they contain, and its indexing vocabulary consists of about one hundred and fifty features. Since, in addition to the use of design goals and outcomes as indices, ArchieTutor's cases are also cross-indexed by domain models and design guidelines, its indexing vocabulary contains about hundred additional features.

Domain Models: The content, representation and organization of domain models in ArchieTutor is similar to that in Archie. As in Archie, domain models in ArchieTutor specify its knowledge of qualitative relations between variables characterizing office buildings. Figure 3 illustrates the presentation of a domain model in ArchieTutor. Archie's models are not accessible to the designer, but ArchieTutor provides the student with access to graphical displays of its models. In addition, the models in ArchieTutor are cross-indexed with the design cases, guidelines and principles in the systems' memory. This enables the student to easily switch from viewing one type of knowledge to another.

Design Principles and Guidelines: The content, representation and organization of design principles and guidelines in ArchieTutor is similar to that in AskJef. Design principles in ArchieTutor are "rational principles" that relate user requirements with the functional characteristics of a building. The six principles in ArchieTutor cover plan and lighting layouts in office buildings. As in AskJef, design guidelines in ArchieTutor operationalize the design principles: they specify the ways in which principles can be achieved. Since a design principle can be achieved in several ways, the system contains twenty-four guidelines that illustrate how to achieve its six principles. These guidelines are organized around the principles they help to achieve, and are cross-indexed with the design cases and domain models in the system's memory. Both the design principles and guidelines are represented in the form of associative rules. Figure 4 illustrates the presentation of a guideline in ArchieTutor.

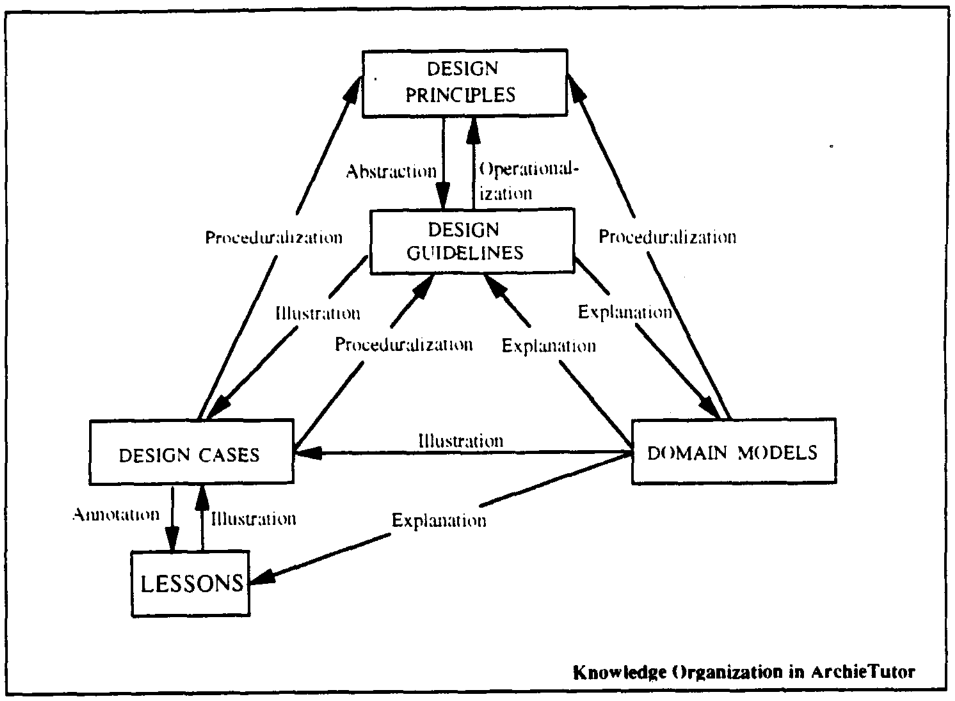
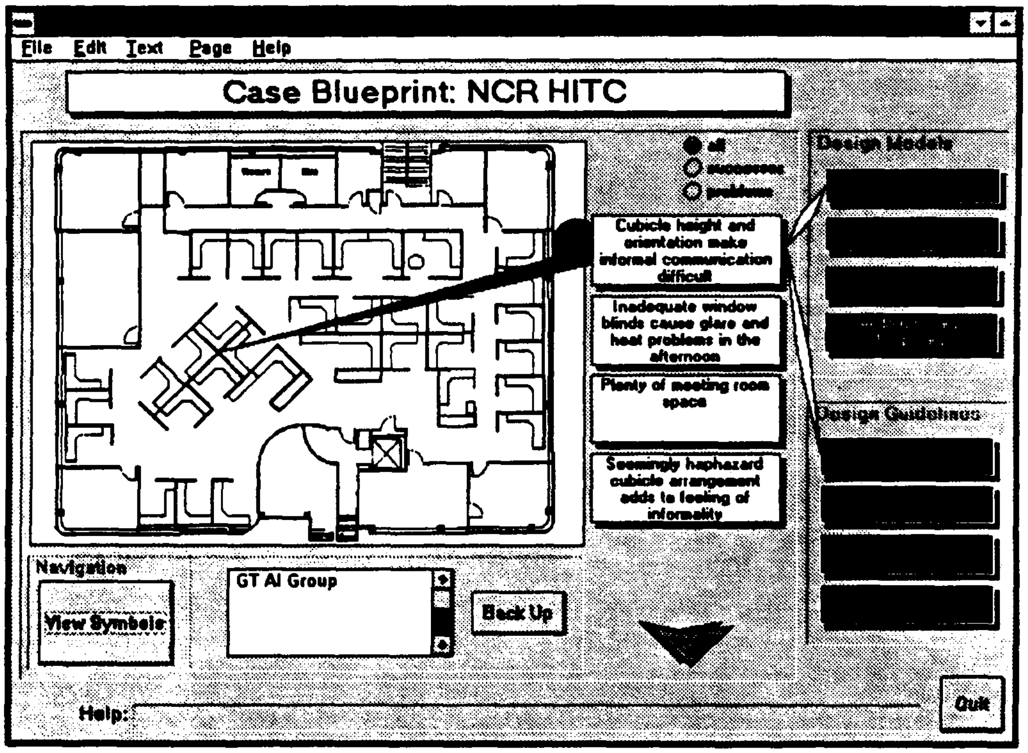


Figure 1: Knowledge organization in ArchieTutor



Case Blueprint: NCR HITC

Cubicle height and orientation make informal communication difficult

Inadequate window blinds cause glare and heat problems in the afternoon

Plenty of meeting room space

Seemingly haphazard cubicle arrangement adds to feeling of informality

Figure 2: Display of a design case in ArchieTutor

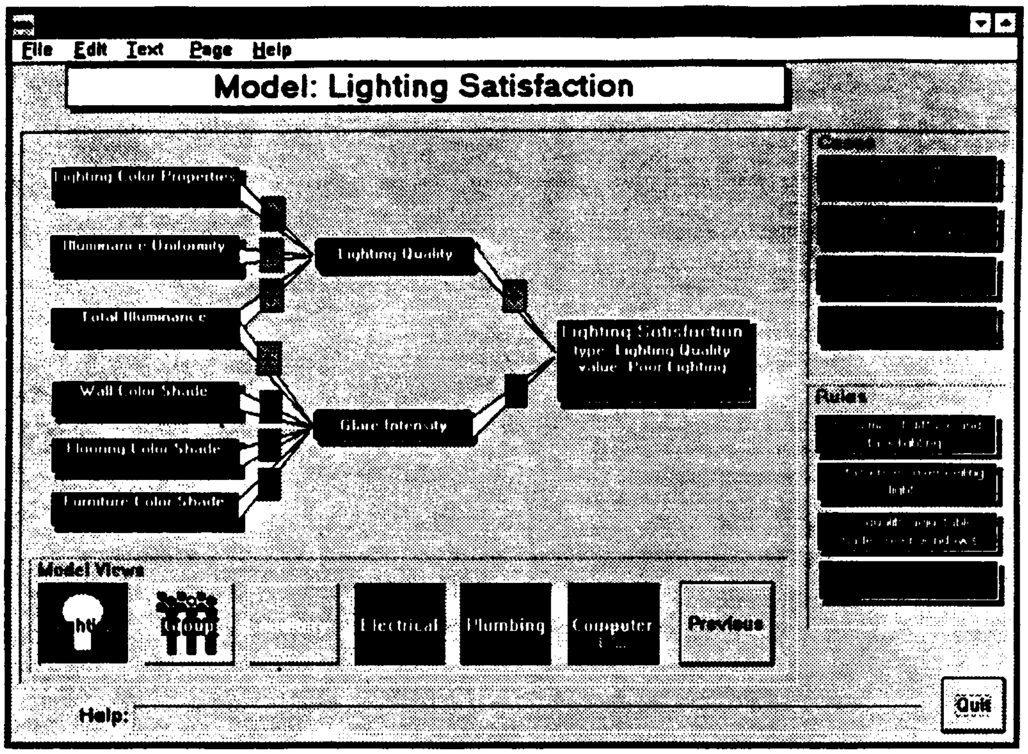
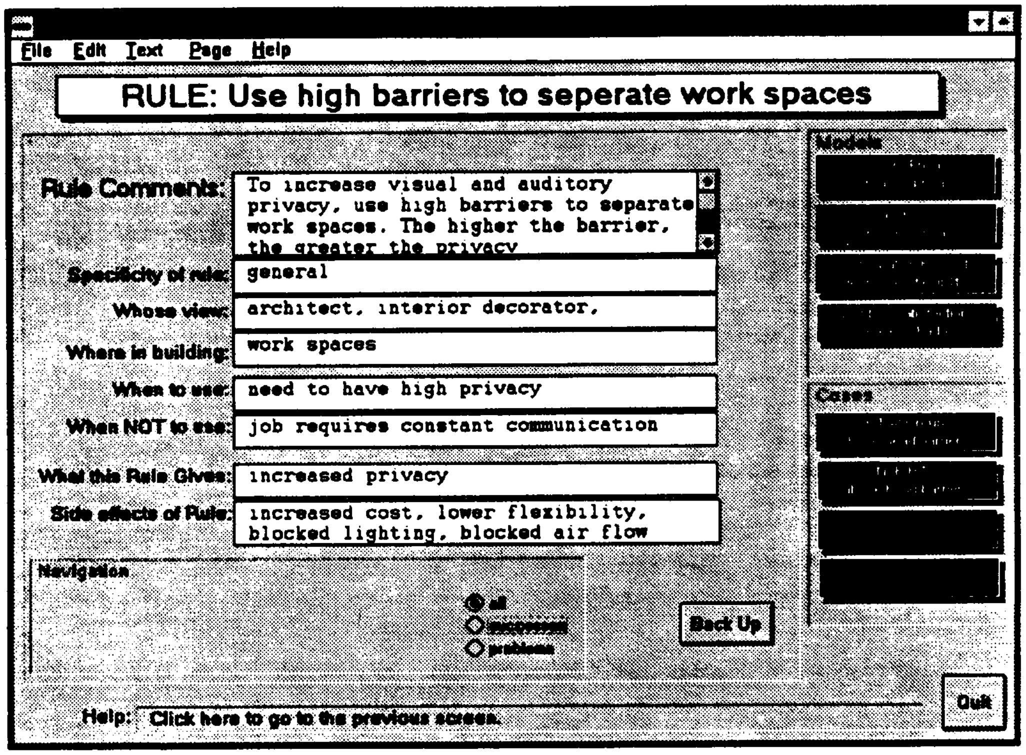


Figure 3: Display of a domain model in ArchieTutor



RULE: Use high barriers to seperate work spaces

Rule Comments: To increase visual and auditory privacy, use high barriers to separate work spaces. The higher the barrier, the greater the privacy

Specificity of rule: general

Whose view: architect, interior decorator,

Where in building: work spaces

When to use: need to have high privacy

When NOT to use: job requires constant communication

What this Rule Gives: increased privacy

Side effects of Rule: increased cost, lower flexibility, blocked lighting, blocked air flow

[✓ all] successes problems

Help: Click here to go to the previous screen.

Figure 4: Display of a design principle/guideline in ArchieTutor

Summary

ArchieTutor provides an architectural student with access to a library of past design cases, and enables the student to retrieve and browse through cases that are similar to a given design problem. The cases are annotated with design lessons learned from them. Further, the cases are explicitly related to relevant design principles and guidelines, and domain models. The system uses these principles, guidelines and models to explain, justify or critique a design choice, and to elaborate on a design lesson. It also provides the student access to other cases that may illustrate a given principle, guideline or model. In this way, it helps the student to discover (i) the structure of the design domain, and of design problems and their solutions in the domain, and (ii) some of the knowledge sources and skills useful in generating design concepts and critiquing design solutions. This form of "subversive tutoring" does not force the student to learn about the domain but invites the student to explore the domain in the context of solving a design problem, and presents knowledge in such a way that it is likely to be remembered. We are presently designing a set of experiments for evaluating ArchieTutor in a classroom setting.

References

[Barber *et al.* 1992] J. Barber, S. Bhatta, A. Goel, M. Jacobson, M. Pearce, L. Penberthy, M. Shankar, R. Simpson, and E. Stroulia. AskJef: Integration of Case-Based and Multimedia Technologies for Interface Design Support. In *Proc. Second International Conference on AI in Design*, Pittsburgh, June 1992, pp. 457-476, Kluwer Academic.

[Goel *et al.* 1991] A. Goel, J. Kolodner, M. Pearce, R. Billington, and C. Zimring. A Case-Based Tool for Conceptual Design Problem Solving. *Proc. Third DARPA Workshop on Case-Based Reasoning*, Washington D.C., May 1991, pp. 109-120, Morgan Kaufmann.

[Kolodner 1991] J. Kolodner. Improving Human Decision Making through Case-Based Decision Aiding. *AI Magazine*, 12(2):52-68, Summer 1991.

[Pearce *et al.* 1992] M. Pearce, A. Goel, J. Kolodner, C. Zimring, L. Sentosa, and R. Billington. Case-Based Design Support: A Case Study in Architectural Design. *IEEE Expert*, 7(5):14-20, October 1992.